

Java con ChatGPT



ECOE
EDICIONES

Paraninfo

Bruce Hopkins

JAVA CON ChatGPT

Hopkins, Bruce, autor

Java con ChatGPT. / Bruce Hopkins -- Primera edición. -- Bogotá: Ecoe Ediciones, Ediciones Paraninfo, 2026.

166 páginas. 17x24 cm

(Lenguajes de programación y extensión/scripting: general. Lenguaje natural y traducción automática)

Incluye datos curriculares del autor -- Incluye referencias bibliográficas.

ISBN 978-958-508-927-3 (impreso) 978-958-508-929-7 (pdf) 978-958-508-928-0 (digital)

1. Inteligencia artificial -- Aprendizaje automático 2. Procesamiento del lenguaje natural 3. Interfaces de programación de aplicaciones (API) 4. Java (Lenguaje de programación) 5. Chatbots 6. Modelos del Lenguaje -- ChatGPT I. Hopkins, Bruce, autor

CDD: 006.31 Ed. 23

Nohora Alvarado



Área 1: Lenguajes de programación y extensión/
scripting: general

Área 2: Lenguaje natural y traducción
automática

Clasificación Thema: UMX - UYQL



ECOE
EDICIONES

Paraninfo

© Bruce Hopkins

© Ediciones Paraninfo, SA

info@paraninfo.es

www.paraninfo.es

Teléfono: (+34) 914 463 350

Calle José Abascal 41,

Oficina 701. 28003

Madrid, España

© Ecoe Ediciones S.A.S.

info@ecoeediciones.com

www.ecoeediciones.com

Carrera 19 # 63 C 32

Teléfono: (+57) 321 226 46 09

Bogotá, Colombia

♦ Cita sugerida:

Hopkins, B. (2026). *Java con ChatGPT*.

Ecoe Ediciones; Paraninfo

Primera edición: Bogotá, 2026

ISBN: 978-958-508-927-3

e-ISBN (PDF): 978-958-508-929-7

e-ISBN (ePub): 978-958-508-928-0

Directora editorial: Ana María Rueda G.

Coordinadora de producción editorial:

Alejandra Rondón Forero

Carátula: Natalia Herrera

Impresión: Carvajal Soluciones de

Comunicación S.A.S. Carrera 69 #15-24

Esta publicación contribuye al logro de los Objetivos de Desarrollo Sostenible 4 y 10, promoviendo una educación de calidad y la reducción de las desigualdades.



La reproducción total de esta obra, ya sea en formato físico o digital, está estrictamente prohibida sin la autorización expresa del titular de los derechos. Asimismo, cualquier reproducción parcial de este libro, con o sin fines comerciales, en formato físico o digital, requiere la autorización previa.

Contenido

I. PRESENTACIÓN DE CHATGPT PARA DESARROLLADORES DE JAVA	1
1.1. ¿A quién va dirigido este libro?	2
1.2. ¡Descarga el código ahora!	2
1.3. Entonces, ¿qué es exactamente ChatGPT y por qué necesito usar las API de OpenAI?	2
1.4. <i>Regex</i> versus ChatGPT. ¡A luchar!	5
1.5. Desaprender algunas palabras para saber más sobre la API de ChatGPT.....	7
1.5.1. Modelos. ¿Modelos? ¡¡¡Modelos!!!	7
1.5.2. Al hablar de <i>tokens</i> hay que pensar en StringTokenizer y no en los <i>tokens</i> de acceso	10
1.5.3. La temperatura tiene que ver con la creatividad	11
1.6. Primeros pasos con OpenAI Playground	12
1.6.1. ¡Pruébalo ahora! Experimenta con el rol de System	15
1.7. Conclusión	16
2. USO DE CHATGPT COMO COMPAÑERO PROGRAMADOR DE JAVA	17
2.1. Creación de tu primera aplicación Java ChatGPT: ListModels.java	18
2.1.1. List Models Endpoint	18



- 2.1.2. Chat Endpoint 23
- 2.1.3. ¿Cuántos *tokens* hay en mi mensaje? 31
- 2.2. Creación de la próxima aplicación Java: ChatGPTClient.java 32
- 2.3. Conclusión 38

- 3. USO DE LA IA EN LA EMPRESA. RESUMIDOR DE TEXTO PARA MENSAJES DE SLACK..... 39**
 - 3.1. ¿Qué es la *prompt engineering*? 40
 - 3.2. Actualización de ChatGPTClient.java (y de las clases relacionadas) con el patrón builder 40
 - 3.3. ChatGPT está aquí para quitarle el trabajo a todo el mundo (no realmente) 45
 - 3.3.1. Un problema verbal real: la atención al cliente de una empresa de software 45
 - 3.3.2. *Prompt engineering* 101: resumen de textos 48
 - 3.3.3. Hablemos de la verdadera *prompt engineering* 54
 - 3.4. Registro de una aplicación Slack Bot 55
 - 3.4.1. Especificar lo que el *bot* puede (y no puede) hacer estableciendo el alcance 57
 - 3.4.2. Cómo encontrar el ID de un canal 62
 - 3.4.3. Cómo utilizar la aplicación *bot* de Slack para capturar automáticamente los mensajes de un canal 62
 - 3.5. Ejercicios 69
 - 3.6. Conclusión 70

- 4. MULTIMODAL AI: CREACIÓN DE UN VISUALIZADOR DE PODCASTS CON WHISPER Y DALL-E 3..... 71**
 - 4.1. Presentación del modelo Whisper de OpenAI 73
 - 4.2. Creación de una aplicación para dividir archivos de audio: AudioSplitter.java 77
 - 4.3. Creación del transcriptor de audio: WhisperClient.java 81
 - 4.4. Divertirse un poco y probar cosas con un *podcast* 85
 - 4.5. Ir a meta: *prompt engineering* GPT-4 para escribir un *prompt* para DALL-E 88
 - 4.5.1. Create Image Endpoint 89



4.6. Creación del generador de imágenes: DALLEClient.java.....	92
4.7. DALL-E <i>prompt engineering</i> y mejores prácticas.....	95
4.8. Ejercicios	97
4.9. Conclusión	97

5. CREACIÓN DE UN <i>BOT COMMUNITY MANAGER</i> AUTOMATIZADO CON DISCORD Y JAVA	99
5.1. Elegir Discord como plataforma de tu comunidad.....	100
5.2. Crear un <i>bot</i> más avanzado que nuestro <i>bot</i> de Slack	101
5.2.1. Crear un <i>bot</i> más avanzado que el típico <i>bot</i> de Discord ...	101
5.2.2. Comprender las funciones de los <i>bots</i>	101
5.3. Ejemplo de banco: Banco dels Ladrones	102
5.3.1. Lo primero es lo primero: creación de un servidor Discord propio	102
5.3.2. Creación del canal de preguntas y respuestas.....	104
5.4. Registro de una nueva <i>app bot</i> con Discord	104
5.5. Información general sobre el <i>bot</i>	106
5.6. Parámetros OAuth2 para el <i>bot</i>	106
5.7. Invitar al <i>bot</i> al servidor propio.....	108
5.8. Configuración de las intenciones de la puerta de enlace y obtención del <i>token</i> de ID de Discord para el <i>bot</i>	109
5.9. Creación de una aplicación robot de preguntas y respuestas en Java para responder a las preguntas de un canal	111
5.9.1. Configuración de las dependencias.....	111
5.9.2. Creación del primer <i>bot</i> de Discord: TechSupportBotDumb.java	112
5.9.3. Expresión <i>lambda</i> para simplificar el código	115
5.9.4. Manejo de mensajes enviados al servidor de Discord	116
5.9.5. Ejecución del primer <i>bot</i> de Discord: TechSupportBotDumb.java	117
5.10. Agilizar el proceso de registro de nuestra próxima aplicación <i>bot</i> de Discord con Discord.....	118
5.10.1. Registro de una nueva <i>app bot</i> de Discord con Discord... ..	118
5.10.2. Información general sobre el <i>bot</i>	118
5.10.3. Parámetros OAuth2 para el <i>bot</i>	119



- 5.10.4. Invitar al *bot* al servidor propio 119
- 5.10.5. Configuración de las intenciones de la puerta de enlace y obtención del *token* de ID de Discord para el *bot*. 119
- 5.10.6. Creación del siguiente *bot* de Discord:
ContentModeratorBotDumb.java 119
- 5.10.7. Manejo de mensajes enviados al servidor de Discord 122
- 5.10.8. Ejecución del segundo *bot* de Discord:
ContentModeratorBotDumb.java 122
- 5.11. Ejercicios 123
- 5.12. Conclusión 123

6. INTELIGENCIA PARA LOS *BOTS* DE DISCORD PARTE I: USO DE CHAT ENDPOINT PARA PREGUNTAS Y RESPUESTAS..... 125

- 6.1. Hacer TechSupportBot.java más inteligente. 126
- 6.2. Cambios importantes que tener en cuenta respecto a la versión anterior del *bot* de soporte técnico. 131
- 6.3. Actualizaciones del método onMessageReceived(). 132
- 6.4. Análisis de ChatGPTClientForQAandModeration.java. 133
- 6.5. Uso de JSONPath para extraer contenido rápidamente en archivos JSON. 137
- 6.6. Ejecución de nuestro robot inteligente de preguntas y respuestas: TechSupportBot.java 137
- 6.7. Tenemos un logro monumental... con un pequeño defecto. 139
- 6.8. Actualización de systemMessage para ChatGPT. 140
- 6.9. Conclusión 142

7. INTELIGENCIA PARA LOS *BOTS* DE DISCORD PARTE II: USO DE CHATS Y MODERATIONS ENDPOINT PARA LA MODERACIÓN 143

- 7.1. Moderations Endpoint 145
 - 7.1.1. Creación de la solicitud. 145
 - 7.1.2. Creación de la moderación (JSON). 145
 - 7.1.3. Manejo de la respuesta JSON. 146
 - 7.1.4. Moderation (JSON). 148



7.2. Creación de un cliente para Moderations Endpoint: ModerationClient.java	150
7.3. Hacer que ContentModeratorBot.java sea más inteligente	154
7.4. Cambios importantes que tener en cuenta respecto a la versión anterior del <i>bot</i> moderador de contenidos.....	158
7.5. Actualizaciones del método onMessageReceived().....	159
7.6. Ejecución del <i>bot</i> moderador de contenidos inteligente: ContentModeratorBot.java	160
7.7. Ejercicios	161
7.8. Conclusión	162

Apéndice.

MODELOS DE OpenAI.....	163
------------------------	-----



1

PRESENTACIÓN DE CHATGPT PARA DESARROLLADORES DE JAVA

En este capítulo vamos a sumergirnos y explicar algunos términos con los que quizás no estés familiarizado, para luego ir directamente al mundo de ChatGPT. Playground es esencialmente una interfaz web para desarrolladores que se basa en la API REST de ChatGPT. Por lo tanto, una vez que estés familiarizado con Playground, las llamadas y respuestas de la API de ChatGPT las sentirás como una segunda naturaleza.

CONTENIDO

- 1.1. ¿A quién va dirigido este libro?
- 1.2. ¡Descarga el código ahora!
- 1.3. Entonces, ¿qué es exactamente ChatGPT y por qué necesito usar las API de OpenAI?
- 1.4. *Regex versus* ChatGPT. ¡A luchar!
- 1.5. Desaprender algunas palabras para saber más sobre la API de ChatGPT
- 1.6. Primeros pasos con OpenAI Playground
- 1.7. Conclusión





1.1. ¿A quién va dirigido este libro?

En primer lugar, este libro está dirigido a desarrolladores de Java que no tienen ninguna formación o experiencia en inteligencia artificial, procesamiento del lenguaje natural, aprendizaje automático o aprendizaje profundo. Es posible que hayas oído hablar del término «modelo de lenguaje», pero voy a suponer que no es un término que uses todos los días.

En segundo lugar, es posible que estés familiarizado con (o hayas probado) ChatGPT, pero no entiendes muy bien cómo funciona todo junto, por lo que no estás seguro de cómo empezar a utilizar Java y ChatGPT juntos mediante programación para «habilitar con IA» tus propias aplicaciones y servicios.

NOTA

Aunque ChatGPT es un nombre familiar, OpenAI, la empresa que está detrás, carece de un amplio reconocimiento y no es tan ampliamente conocida. Entonces, aunque este libro trata sobre cómo usar ChatGPT programáticamente dentro de tus aplicaciones Java, las API que usaremos son oficialmente las API REST de OpenAI. Por lo tanto, usaremos los términos «API de ChatGPT» y «API de OpenAI» como sinónimos.

1.2. ¡Descarga el código ahora!

Este libro está repleto de ejemplos y código de ejemplo, así que ¿por qué molestarse en escribir todo tú mismo? A continuación se muestra la dirección URL del repositorio de GitHub para acceder al código fuente.



<https://github.com/JavaChatGPT>

1.3. Entonces, ¿qué es exactamente ChatGPT y por qué necesito usar las API de OpenAI?

En pocas palabras, ChatGPT es un grupo de **modelos de lenguaje** (que son el resultado del entrenamiento de algoritmos de aprendizaje profundo) desarrollados por OpenAI y que son capaces de generar respuestas de texto similares a las humanas basadas en **prompts**. Entonces, ¿qué es un *prompt*? Un *prompt* es simplemente el texto de entrada que le das a ChatGPT. Así que comencemos dándole a ChatGPT un *prompt* para preguntar qué tan bien está familiarizado con los patrones de diseño de Java.



Listado 1.1.

PROMPT. Un sencillo mensaje con ChatGPT sobre los patrones de diseño de Java

Describir 3 patrones de diseño comúnmente utilizados por los desarrolladores Java y por qué esos patrones son importantes.

Si has tenido la oportunidad de trabajar en una empresa utilizando Java de forma profesional, probablemente hayas recibido una pregunta muy similar a la siguiente en una entrevista. La respuesta de ChatGPT la encontrarás a continuación, en el Listado 1.2.

Listado 1.2.

RESPUESTA de ChatGPT al *prompt* planteado



¡Ciertamente! Los patrones de diseño son soluciones reutilizables a problemas comunes que surgen en el diseño de software. Son las mejores prácticas que los desarrolladores experimentados han descubierto y perfeccionado con el tiempo. Los desarrolladores de Java utilizan con frecuencia patrones de diseño debido a la naturaleza orientada a objetos del lenguaje. Aquí hay tres patrones de diseño comúnmente utilizados en Java:

1. Patrón Singleton:

- Propósito: este patrón garantiza que una clase tenga solo una instancia y proporciona un punto de acceso global a esa instancia. Es especialmente útil cuando se desea controlar el acceso a los recursos compartidos, como los administradores de configuración o los grupos de conexiones.
- Importancia: el patrón Singleton es importante porque proporciona un mecanismo controlado para evitar la creación de varias instancias de una clase, lo que puede ahorrar memoria y sincronizar el acceso a los recursos. Además, garantiza que las operaciones que usan el *singleton* usen la misma instancia y los mismos datos, lo cual es fundamental en escenarios en los que se requieren acciones coordinadas o de estado coherentes.

2. Patrón de observador:

- Propósito: este patrón define una dependencia de uno a varios entre objetos. Cuando un objeto (conocido como sujeto u observable) cambia de estado, todos sus dependientes (observadores) son notificados y actualizados automáticamente. Es el principio subyacente detrás de la arquitectura Modelo-Vista-Controlador (MVC), donde el modelo es el sujeto y las vistas son observadores.



- Importancia: el patrón Observer es vital para construir sistemas modulares y desacoplados. Los cambios en una parte del sistema no afectan directamente a otras, lo que facilita el mantenimiento y la escalabilidad. Este patrón también facilita la adición de nuevos observadores sin modificar el código existente, lo que mejora la extensibilidad del sistema.
3. Patrón de fábrica:
- Propósito: el patrón Factory proporciona una interfaz para crear instancias de una clase, pero depende de las subclasses derivadas para implementar el método y determinar qué clase instanciar. Promueve el acoplamiento flexible al eliminar la necesidad de enlazar clases específicas de la aplicación en el código. En su lugar, el código interactúa solo con la interfaz resultante o la clase abstracta, por lo que el sistema puede evolucionar sin alterar el código de llamada.
 - Importancia: con el patrón Factory, un sistema puede ser más flexible y escalable. Si es necesario introducir un nuevo producto o modificar uno existente, se puede hacer sin alterar el código existente. Encapsula el proceso de creación de objetos y lo separa de la lógica principal del programa, promoviendo el principio de responsabilidad única. Esto garantiza que una clase solo sea responsable de un motivo para cambiar.

Todos estos patrones, y muchos más, proporcionan un vocabulario para los desarrolladores, lo que facilita la comunicación de diseños y conceptos de software complejos. Incorporan soluciones a problemas recurrentes y pueden acelerar significativamente el proceso de desarrollo, reducir los errores y mejorar la calidad del código.

¡No está mal! ChatGPT no solo pudo proporcionar una lista de patrones de diseño comunes utilizados por los desarrolladores de Java, sino también explicar razones válidas sobre por qué son importantes, todo confeccionado en lenguaje natural.

Entonces, al final del día, ¿qué significa esto para los desarrolladores de Java? En realidad es un gran problema. Esto se debe a que los desarrolladores ahora pueden crear aplicaciones que pueden tener algún tipo de «comprensión» del texto en lenguaje natural. Antes de la llegada de herramientas de IA como ChatGPT, los desarrolladores solían tener que crear expresiones regulares (también llamadas *regex*) para realizar una coincidencia rudimentaria de caracteres y patrones de cadenas dentro de un cuerpo de texto. Sin embargo, la coincidencia de patrones y la comprensión del lenguaje natural son dos cosas totalmente diferentes.



1.4. *Regex versus ChatGPT. ¡A luchar!*

Estoy completamente convencido de que todos los programadores, en algún momento de su vida, han conocido a «alguien» que resulta ser un experto en escribir expresiones regulares. Las expresiones regulares son excelentes porque sirven para poder analizar grandes cantidades de texto con el fin de encontrar patrones dentro del texto mediante programación.

NOTA

Si ya sabes que una expresión regular no puede realizar la comprensión del lenguaje natural o el análisis de sentimientos, no dudes en omitir esta sección.

Sin embargo, una de las mayores desventajas de las expresiones regulares es que una vez que se han escrito son extremadamente difíciles de leer (en mi opinión, incluso para el desarrollador que las escribió originalmente).

Así que veamos cómo se comporta una expresión regular frente a ChatGPT, que tiene capacidades de procesamiento del lenguaje natural (NLP) y comprensión del lenguaje natural (NLU).

El Listado 1.3 es la historia de una situación impracticablemente triste. Sin embargo, lleva a casa el punto de que, aunque las expresiones regulares se pueden usar para encontrar palabras y frases dentro de un cuerpo de texto, no es posible usarlas para proporcionar ningún tipo de NLU.

Listado 1.3.

Sadstory.txt. Una triste historia sobre un niño que no comía helado

Pregunta de análisis n.º 1: ¿Quién no compró helado y por qué?

Ahora analicemos esto un poco y hagámonos algunas preguntas. En primer lugar, ¿quién no compró helado y por qué? La respuesta obvia es que Steve no comió ningún helado debido a su intolerancia a la lactosa. Sin embargo, dado que la historia no dice directamente que Steve no compró helado, no hay forma de que una expresión regular coincida con un patrón de texto en la historia.

La expresión regular podría buscar fragmentos clave como «no tuvo», «no compró helado» o los nombres de los niños. Sin embargo, solo podría dar una respuesta basada en la presencia de estos patrones. Por ejemplo, si la expresión regular coincide con el patrón «no tuvo» o «no compró helado» con el nombre de Steve, podría mostrar el resultado del patrón de texto. Sin embargo, ciertamente no podría explicar por qué Steve fue el que no comió helado ni proporcionar ningún razonamiento específico del contexto.



Ahora, proporcionemos la misma historia a ChatGPT y hagamos la pregunta: «¿Quién no compró helado?». El Listado 1.4, que se encuentra a continuación, toma nuestra pregunta y la historia anterior y las pone juntas como un *prompt*.



Usando la información de la historia a continuación, ¿quién no compró helado y por qué?

###

En la ciudad de Buttersville, EE.UU., en la calle Milkmaid, hay un grupo de tres amigos: Marion Yogurt, Janelle de Queso y Steve Cheeseworth III. En un caluroso día de verano, escucharon la música de un camión de helados y decidieron comprar algo para comer.

A Marion le gustan las fresas, Janelle prefiere el chocolate y Steve es intolerante a la lactosa. Ese día, solo dos niños comieron helado y uno de ellos compró una botella de agua a temperatura ambiente. El camión de helados estaba completamente abastecido con los sabores típicos de los helados.

Listado 1.4.

PROMPT. La triste historia puesta en un *prompt*

Observa cómo es una práctica recomendada proporcionar algún tipo de separación, en este caso, «###», al crear un mensaje (como el anterior) en el que hay instrucciones y datos juntos. Más adelante, cuando empecemos a utilizar Playground o Java para invocar mediante programación la API de ChatGPT, verás que hay una forma mucho mejor de proporcionar esa separación.

Entonces, después de enviar el mensaje, ChatGPT te proporcionará una respuesta como la del Listado 1.5.

Listado 1.5.

RESPUESTA de ChatGPT a la pregunta de análisis n.º 1



Según la información proporcionada, Steve es intolerante a la lactosa y, por lo tanto, no puede comer helado. Por lo tanto, Steve es el que no compró ningún helado.

Como puedes ver, ChatGPT puede utilizar NLP y NLU y, por lo tanto, puede «comprender» (es decir, de manera artificialmente inteligente) el contexto del escenario. Puede interpretar las relaciones entre los niños, sus preferencias y la intolerancia a la lactosa de Steve. Es capaz de entender que los apellidos de los niños, el nombre de la calle y el nombre de la ciudad son los nombres de los productos lácteos, pero obviamente son irrelevantes para el asunto en cuestión.



Pregunta de análisis n.º 2: ¿Qué niño probablemente se quedó triste?

Ahora, para demostrar aún más el punto de que una expresión regular no puede proporcionar ningún tipo de NLP o NLU, usaremos un nuevo término, denominado **análisis de sentimiento**. Por lo tanto, cuando el camión se hubo alejado, ¿qué niño se quedó triste?

Dado que el cuento no menciona ninguno de los sentimientos o emociones de los niños, no hay un patrón de texto que permita que cualquier expresión regular devuelva una coincidencia.

Sin embargo, si le hiciste la misma pregunta a ChatGPT, te devolverá una respuesta como la del Listado 1.6.

Listado 1.6.

RESPUESTA de ChatGPT a la pregunta de análisis n.º 2



Dado que Steve es intolerante a la lactosa y no puede comer helado, sería el niño que se quedaría triste porque no pudo disfrutar del helado como Marion y Janelle.

Por lo tanto, ChatGPT es capaz de comprender el escenario, razonar a través de la información y proporcionar una respuesta correcta junto con una «explicación» de esa respuesta.

1.5. Desaprender algunas palabras para saber más sobre la API de ChatGPT

En primer lugar, antes de empezar a trabajar con las API de ChatGPT y OpenAI, hay algunas palabras y términos con los que deberías estar familiarizado; de lo contrario, las cosas no tendrán mucho sentido. Así que debes asegurarte de tener clara la definición de modelos, *prompts*, *tokens* y *temperature* para utilizar ChatGPT mediante programación.

1.5.1. Modelos. ¿Modelos? ¡¡¡Modelos!!!

Como desarrollador de Java, al escuchar el término «modelo» es posible que pienses inmediatamente en la programación orientada a objetos y la representación de entidades del mundo real en tu clase de Java, ¿verdad? Por ejemplo, piensa en el término «modelo de objetos». Además, si alguna vez has trabajado con cualquier tipo de base de datos, entonces el término «modelo» también puede evocar en tu mente la idea de la representación de los datos y sus relaciones en tu base de datos. Por ejemplo, piensa en el término «modelo de datos».